Precise Localization using Sweeps in Sparse Networks

Goldenberg, D. K.; Bihler, P.; Cao, M.; Fang, J.; Anderson, B. D. O.; Morse, A. S.; Yang, Y. R.

Published in:
Proceedings of the 12th ACM Annual International Conference on Mobile Computing and Networking (MobiCom)

Localization in Sparse Networks using Sweeps

David K. Goldenberg* Pascal Bihler* Ming Cao† Jia Fang†
Brian D.O. Anderson§ A. Stephen Morse*† Y. Richard Yang*
Yale University Departments of Computer Science* and Electrical Engineering†
Australian National University and National ICT Australia Ltd.§
{david.goldenberg,pascal.bihler,ming.cao, jia.fang, as.morse, yang.r.yang}@yale.edu
brian.anderson@nicta.com.au

ABSTRACT

Determining node positions is essential for many next-generation network functionalities. Previous localization algorithms lack correctness guarantees or require network density higher than required for unique localizability. In this paper, we describe a class of algorithms for fine-grained localization called *Sweeps*. Sweeps correctly *finitely localizes* all nodes in *bilateration* networks. Sweeps also handles angle measurements and noisy measurements. We demonstrate the practicality of our algorithm through extensive simulations on a large number of networks, upon which it consistently localizes one-thousand-node networks of average degree less than five in less than two minutes on a consumer PC.

Categories and Subject Descriptors: C.2.1 [Computer Communication Networks]: Network Architecture and Design – *Wireless communication*; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Performance, Design.

Keywords: Localization, Sweeps, Global Rigidity, Controlled Mobility

1. INTRODUCTION

Determining node positions or possible positions is an essential requirement for many next-generation network functionalities. For example, in crisis response, it is important to know the precise position or possible positions of an emergency in order to take prompt action. For some inventory applications or ubiquitous computing, it is necessary to precisely identify one particular item out of a large number of items in close proximity. Although the importance of precise localization has long been well-recognized, no entirely satisfactory solution yet exists for the anticipated networks of thousands of resource-constrained, and possibly mobile nodes.

Naive approaches are easily seen to be inadequate. While it may be possible to manually furnish each node with its position in small and static networks, this approach is clearly infeasible in the envisioned large-scale networks. Another straightforward approach is to equip each node with GPS, enabling it to determine its position by communicating with GPS satellites. However, there are several potential problems with this approach. First, GPS hardware-requirements may be excessive for resource-constrained nodes in large-scale networks. Next, GPS-reliant localization will not be robust in the presence of obstructions such as dense foliage and tall buildings blocking communication with satellites, and has great difficulty operating indoors or underground. Another potential pitfall in designing a GPS-reliant localization layer is that it will be dependent on the GPS infrastructure, which may come under attack or be made unavailable by its owner.

The difficulties in the obvious approaches have led researchers to address the problem in alternative technological settings. A promising class of approach for precise localization is *fine-grained localization* (*e.g.*, [1, 7, 8, 11, 13, 21, 28, 29, 30, 31, 32, 34, 35, 36, 39]). In such approaches, only some of the network nodes called *beacons* or *anchors* are endowed with their positions through GPS or manual configuration, and all nodes measure the distance between themselves and nearby nodes using hardware ranging techniques. The essential aim of fine-grained localization is to propagate the knowledge of the positions of only a few nodes to the positions of many using relationships in position expressed by pairwise distance information.

Fine-grained localization algorithms can be broadly classified into two categories: the global approaches, which localize all nodes simultaneously (*e.g.*, [8]), and the sequential approaches, which localize nodes in some order (*e.g.*, [28, 35]). A representative global approach is semi-definite programming (SDP) [8, 7]. This approach works well for a dense network. However, it may generate faulty positions in sparse networks where global incorrect flip configurations cannot be escaped. Another technique to compute all node positions simultaneously is multi-dimensional scaling (MDS) (*e.g.*, [21, 25, 36]). However, the quality of MDS position estimates depends crucially on the quality of the estimate of the complete distance matrix. The problem of producing good estimates of the complete distance matrix for non-convex network deployments is still an unsolved problem. Although important work was done by Lim and Hou [24] addressing this issue, since their approach utilizes anchors, the problem still stands for cases in which anchors are not distributed uniformly, or there are no anchors at all. Overall, the global approaches treat all nodes equally without identifying correctly localizable nodes. Given the inherent NP-hardness of localization [4], it is unlikely that the global approaches can avoid large errors for some nodes. Such errors can happen particularly in sparse networks or dense networks with sparse sub-regions. However, as the authors of [28] pointed out, "*for many applications, missing localization information for a known set of nodes is preferential to incorrect information for an unknown set*."

One way to guarantee correctness is to localize nodes sequen-

MobiCom'06, September 23–26, 2006, Los Angeles, California, USA.

tially. A representative sequential approach is robust quadrilaterals [28]. This approach processes nodes one by one, and in the process, prunes distance measurements that are deemed unreliable. Although this approach ensures the absence of systematic localization errors, it succeeds in localizing only a small portion of potentially localizable nodes. It fails on sparse networks, and does not succeed in extending a localized region past areas of local sparsity.

In this paper, we study fine-grained localization with the following requirements: 1) compute positions without potential systematic errors; 2) localize uniquely localizable nodes with high probability, even in networks that are not uniformly dense; and 3) for nodes that cannot be uniquely localized but can be localized up to a set of possibilities, output the set of possible positions whenever feasible. It is easy to envision that outputting the set of all possible positions of a node can be useful in many applications. We call this generalized localization objective *finite localization*, as opposed to unique localization, upon which most previous localization techniques focus.

In this paper, we provide a class of simple algorithms referred to as Sweeps which satisfies the above design requirements. In prior work [15, 16], the idea of sweeping through a network in a sequential fashion was proposed and preliminary results were obtained. In this paper, we improve the computational complexity of sweeping using *consistent* position combinations and *shell* sweeps, and extend sweeping to handle angle and noisy measurements. By design, our algorithm does not fall victim to large localization errors due to flip configurations. We prove that our algorithm finitely localizes all nodes in a large class of sparse network called *bilateration networks*. For uniformly random networks, we demonstrate that in the worst case, our algorithm uniquely localizes at least 90% of all uniquely localizable nodes, when the average connectivity degree of the network nodes varies from 3 to 13; except in the transition phase when the average connectivity degree of the network nodes is between 6 and 7.5, it localizes above 95% of all uniquely localizable nodes. As comparison, iterated trilateration sometime localizes only 40% of the uniquely localizable nodes. For regular networks deployed to provide spatial coverage, at average connectivity degree 6, our algorithm localizes 90% of uniquely localizable nodes, while iterated trilateration localizes less than 10%.

The tradeoff for achieving high localizability at low density is that the worst-case time complexity of our algorithm could be exponential in the number of nodes. However, we show that at a given density, typically, the running time of our algorithm grows linearly with increasing number of nodes. It localizes more than 95% of uniquely localizable nodes of a network with one-thousand nodes with average connectivity degree less than five in two minutes on a consumer PC with an Intel CPU of 2.8 GHz. Our algorithm is not essentially centralized, in that it does not require any global information. Thus, our algorithm can be extended to distributed settings. We also extend our algorithm to handle angle and noisy measurements.

As a demonstration on the applicability of our algorithm, we investigate localization in a mobile network in which individual nodes use controlled mobility to optimize spatial coverage. We show that extremely sparse networks with just enough constraints for unique localizability are produced in this setting, and that our algorithm succeeds in localizing these sparse networks feasibly and predictably[1].

The rest of the paper is structured as follows. In Section 2 we give useful terms and definitions, and review background results. In Section 3 we describe our algorithms and in Section 4 we prove its correctness on bilateration networks. In Section 5 we present our simulation results on realistic networks. In this section, we also present our case-study of a coverage-optimizing mobile network and the success of Sweeps Localization on this network class. In Section 6 we cover related work. We conclude and discuss future work in Section 7.

[1]Here predictability is in the sense that we can quickly check whether Sweeps will successfully localize the network without actually performing the localization.

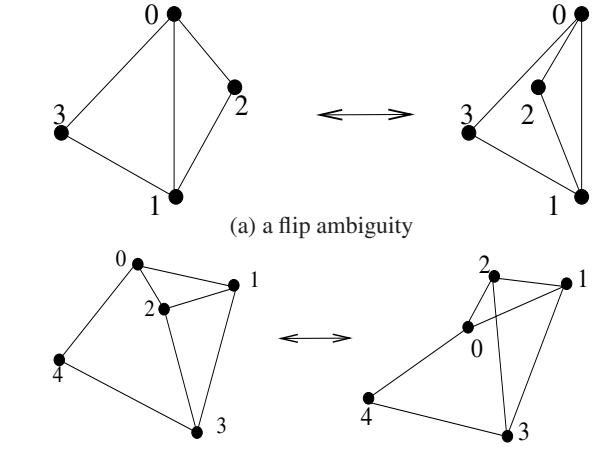
2. BACKGROUND

In this section, we briefly review the theoretical background of localization. For a more thorough exposition of the concepts which follow, we refer the interested reader to [14, 3].

In our network model, we assume that nodes are located at distinct physical locations in some region of space. Let $\mathbb{N}$ be a network in $\mathbb{R}^d$ with nodes labeled 1, 2, ..., n. Let $\pi(i)$ denote the position of node i. Suppose the positions of some nodes are known. The nodes whose positions are known are also called *anchors*.

Below we assume that nodes have some means by which to measure the distance between themselves and their neighboring nodes. Later in this paper, we also consider the case of angle measurements. If the distance between nodes i and j is known, then let d_{ij} denote that distance. Note that the distance between any two anchors is known since the positions of all of the anchors are known. By a *consistent assignment* of the network $\mathbb{N}$ is meant any function $\alpha : \{1, 2, \ldots, n\} \to \mathbb{R}^d$ where $\alpha(a) = \pi(a)$ whenever node a is an anchor, and for all $i, j \in \{1, 2, \ldots, n\}$, $||\alpha(i) - \alpha(j)|| = d_{ij}$ if the distance between nodes i and j is known.

If there exists exactly one consistent assignment of $\mathbb{N}$, we say that the network $\mathbb{N}$ is *uniquely localizable*, or simply *localizable*. A node v of $\mathbb{N}$ is said to be *localizable* if for all consistent assignments α for $\mathbb{N}$, we have that $\alpha(v) = \pi(v)$. There are networks in which the positions of some can be varied continuously while simultaneously satisfying all distance constraints. Such networks have an infinite number of possible position assignments, and are called *flexible*. Networks that are not flexible are called *rigid*. If a network has a finite number of consistent assignments, it is called *finitely localizable*.



(a) a flip ambiguity

(b) a discontinuous flex ambiguity

Figure 1: Rigid networks with flip and discontinuous flex ambiguities.

Rigid networks have a finite number of possible position assignments. Rigidity can be combinatorially characterized generically in the plane by Laman's condition [22], which expresses the well-distributedness of distance constraints over the graph. The multiple possible position assignments of a rigid network in the plane are due to *flip ambiguities* and *discontinuous flex ambiguities*. It is not

known whether or not in 3D there could be other discontinuous ambiguities. In a flip ambiguity, a set of nodes is reflected across a line between a separating pair of nodes. An example is shown in Figure 1(a). A discontinuous flex ambiguity occurs when the network becomes flexible upon removal of a single edge and then subject to a continuous deformation over which at some configuration differing from the original, the removed constraint becomes satisfied and can then be reinserted. An example is shown in Figure 1(b).

We are interested in localization in the *generic* sense, and as such, a graph-theoretic property for *almost* all problem instances. A *multi-point* $p = \{p_1, \ldots, p_n\}$ in d-dimensional space is a set of n points in $\mathbb{R}^d$ labeled $p_1, \ldots, p_n$. A multi-point p is *generic* if the coordinates of points in p are algebraically independent over the rationals. If we assume precise distance measurements, we can neglect such degeneracies as vanishingly unlikely in random networks.

Two multi-points $p = \{p_1, \ldots, p_n\}$ and $q = \{q_1, \ldots, q_n\}$ of n points are *congruent* if for all $i, j \in \{1, \ldots, n\}$, the distance between p_i and p_j is equal to the distance between q_i and q_j. A *point formation* of n points at a multi-point $p = \{p_1, \ldots, p_n\}$ consists of p and a simple undirected graph $\mathbb{G}$ with vertex set $\mathcal{V} = \{1, \ldots, n\}$, and is denoted by $(\mathbb{G}, p)$. If (i, j) is an edge in $\mathbb{G}$, then the *length* of edge (i, j) in the point formation $(\mathbb{G}, p)$ is the distance between p_i and p_j. A network with n nodes is modeled by a point formation $(\mathbb{G}, p)$, where each node corresponds to exactly one vertex of $\mathbb{G}$, and vice versa, with (i, j) being an edge of $\mathbb{G}$ if i and j are distinct and the distance between the corresponding nodes is known, and $p = \{p_1, \ldots, p_n\}$ where p_i is the position of the node corresponding to vertex i. We say that $\mathbb{G}$ is the graph of the network, and p is the multi-point of the network. Since *almost all* multi-points are generic, we have that the multi-points of networks are *almost always* generic. Henceforth, we shall consider mainly networks with generic multi-points. However, degeneracy may become important when distance measurements are imprecise. We will return to this topic in Section 5.

It was shown in [3, 14] that a network is localizable if the *grounded graph* of the network consisting of a vertex for each network node and an edge for every distance measurement and every pair of anchors, is *globally rigid* (triconnected and remains rigid upon removal of any single edge). Specifically, a point formation $(\mathbb{G}, p)$ is *globally rigid* in $\mathbb{R}^d$ if p and q are congruent multi-points in $\mathbb{R}^d$ whenever $(\mathbb{G}, p)$ and $(\mathbb{G}, q)$ have the same edge lengths. A graph $\mathbb{G}$ is said to be *generically globally rigid* in $\mathbb{R}^d$ if $(\mathbb{G}, p)$ is globally rigid in $\mathbb{R}^d$ whenever p in $\mathbb{R}^d$ is generic. There are a number of efficient algorithms for determining if a graph is generically globally rigid in $\mathbb{R}^2$. Since *almost all* multi-points are generic, we have that $(\mathbb{G}, p)$ is globally rigid in $\mathbb{R}^2$ for *almost all* multi-points p in $\mathbb{R}^2$ if $\mathbb{G}$ is generically globally rigid in $\mathbb{R}^2$. A graph that is generically globally rigid in $\mathbb{R}^2$ is said to be *minimally generically globally rigid* in $\mathbb{R}^2$ if the removal of any edge causes the graph to not be generically globally rigid in $\mathbb{R}^2$.

The computational complexity associated with localizing uniquely localizable networks has been shown to be NP-hard even for unit-disk networks [4]. Intuitively, this complexity is linked with exponential growth in possible network configurations due to flip ambiguities present before all constraints are taken into consideration. In spite of the NP-hardness of localization, as we shall see in this paper, there are large classes of networks for which localization is efficiently computable.

One type of efficiently localizable network is trilateration networks. A graph has a *trilateration ordering* with *seeds* v_1, v_2 and v_3 if its vertices can be ordered as $v_1, v_2, v_3, \ldots, v_n$ so that v_1, v_2 and v_3 induce a complete subgraph, and each v_i, $i > 3$, is adjacent to at least three vertices v_j where $j < i$. Graphs with trilateration orderings are called *trilateration graphs* and are generically globally rigid in $\mathbb{R}^2$. We say a network is a *trilateration network* if its graph has a trilateration ordering.

A graph has a *bilateration ordering* with *seeds* v_1, v_2 and v_3 if its vertices can be ordered as $v_1, v_2, \ldots, v_n$ so that v_1 and v_2 are adjacent, and each v_i, $i > 2$, is adjacent to at least two vertices v_j where $j < i$. Graphs with bilateration orderings are called *bilateration graphs*, and a network is called a *bilateration network* if its graph is a bilateration ordering. An example bilateration graph with vertex set $\mathcal{V}$ is a *wheel graph*, if there exists a vertex $v \in \mathcal{V}$ such that v is adjacent to all other vertices in $\mathcal{V}$, and vertices in $\mathcal{V} - \{v\}$ can be ordered as $v_1, \ldots, v_m$ such that each v_i, $i \in \{2, \ldots, m-1\}$, is only adjacent to v_{i-1} and v_{i+1}, and v_1 is adjacent to v_m. Figure 2 shows an example of a wheel graph with 6 vertices.

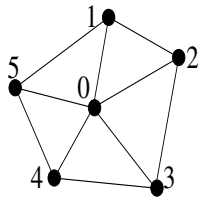


Figure 2: A wheel graph.

In this paper, we say a network is *sparse* if its average node degree is less than 10. Iterative localization algorithms based on trilateration have difficulty localizing sparse networks, but we will show that our algorithm localizes most localizable nodes in sparse networks.

3. PRECISE LOCALIZATION USING SWEEPS

The idea of the Sweeps algorithm is related to simple iterated trilateration. In iterated trilateration, an initial set of three nodes is fixed and used to define a coordinate system. At each stage of the algorithm, there is a set of localized nodes and a set of unlocalized nodes. If an unlocalized node has distance measurements to at least three localized nodes, its position is calculated and it is added to the set of localized nodes. Simple iterated trilateration is sub-optimal in that there are many localizable networks which it cannot localize. Only networks called trilateration networks are completely localized by iterated trilateration [3, 14]. Wheel networks (Figure 2) are an example of a class of localizable but non-trilateration networks [3, 14]. We will see that all generic wheel networks can be localized by the Sweeps algorithm, which we now describe.

3.1 The Basic Shell Sweeps Algorithm

The objective of our Sweeps algorithm is to localize a much larger class of networks than previously possible. In the basic Sweeps algorithm, as in iterated trilateration, an initial set of three nodes is fixed. At each stage of the algorithm, there is a set of *finitely localized nodes* whose positions have been determined up to a finite set of possibilities, and a set of unlocalized nodes. We say that the finitely localized nodes have been "swept". If an unlocalized node has distance measurements to at least two finitely localized nodes, it calculates all possible positions for itself based on the *consistent* combinations of these nodes' positions. Two node positions p_u and p_v are consistent if for every node w whose set of possible positions P_w is used in computing both p_u and p_v, only a single possibility $p_w \in P_w$ is used. In other words, if p_u and p_v both depend on a position of w, they must depend on the same possibility. This is illustrated in Figure 3.

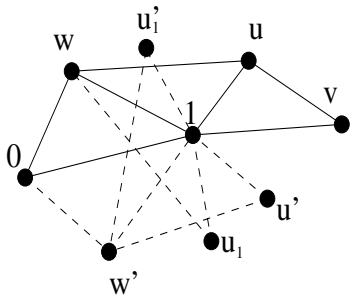


Figure 3: A network illustrating consistency. The sweep starts at nodes 0 and 1. Node w is then finitely localized to possibilities w and w'. Node u is then finitely localized using the unique position of node 1 and each of the two possible positions of node w. Possibilities u and u_1 are derived from w, while u' and u_1' are derived from w'. The position of v is not consistent with u' and u_1' because they depend on w' while v depends on w.

Let us consider a run of Sweeps on the wheel network shown in Figure 2. We use v_i to refer to node i in the figure. We fix the coordinates of v_0, v_1, and v_2 so that $v_0 = (0,0), v_1 = (a,0)$ and $v_2 = (b,c)$ for some $a, c > 0$. Knowledge of the lengths of v_2v_3 and v_0v_3 establishes the position of v_3 with a binary ambiguity. For each of these possible positions for v_3, knowledge of the lengths v_3v_4 and v_0v_4 will establish the position of v_4 with a binary ambiguity, making four possibilities in all. Lastly, we obtain eight possible positions of v_5. For an analogously labelled wheel network of arbitrary size $k+1$, in a similar fashion, we obtain the positions of $v_6, v_7, \ldots, v_k$ with $2^4, 2^5, \ldots, 2^{k-2}$ ambiguities. However, v_k is also connected to v_0 and v_1. Knowledge of the associated lengths resolves the ambiguity in the position of v_k. This in turn allows resolution of the ambiguity in $v_{k-1}, v_{k-2}, \ldots, v_3$, and in this way the unique localization of the network is established.

We have seen that if we run Sweeps on the wheel network, we run into exponential growth in possible node positions. One challenge facing the Sweeps algorithm is to avoid this effect if possible. To this end, we eliminate possibilities as soon as it is possible to do so. After a node is added to the set of swept nodes, all of its distance measurements to other already swept nodes are considered, potentially eliminating some of its possible positions, and some of the possible positions of already swept nodes. Note that there exist examples, such as wheel networks, for which there are no chances to eliminate any possibilities until the very last edge is added.

We further reduce the growth in possible positions by choosing a particular sweep ordering. In the so-called *shell Sweeps*, we perform a breadth-first sweep in which at each stage, all nodes having a distance measurement to at least two already swept nodes are placed earlier in the ordering than all other nodes. In localizing the wheel network of Figure 2, this would result in the order v_0, v_1, v_2, v_3, v_5, v_4, with 1, 1, 1, 2, 2, and 4 respectively being the maximum number of possible positions, instead of the ordering v_0, v_1, v_2, v_3, v_4, v_5 of the naive method, where we have 1, 1, 1, 2, 4, and 8 respective maximum possible node positions. In typical random and regular networks, this approach dramatically reduces the number of possibilities that the algorithm has to maintain.

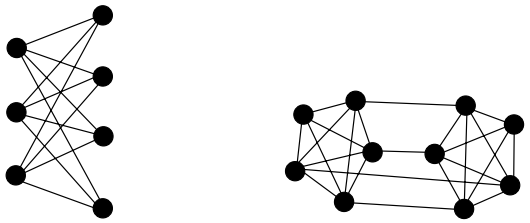
The complete shell sweeps algorithm is shown in Figure 4. The correctness of the algorithm will be analyzed in Section 4.

Sweeps succeeds in finitely localizing all nodes in *bilateration networks* [15, 16], which as we saw in Section 2 are defined analogously to trilateration networks [3, 14]. While bilateration networks are rigid, they are not necessarily globally rigid nor are globally rigid networks necessarily bilateration networks In Figure 5 there are two example of globally rigid graphs that are not bilat-

```
ShellSweep(Node u, Node v, Node w)
    List Order = Order(u, v, w)
    foreach (Vertex x in Order)
        List Ancestors = (Neighbors of x earlier in Order)
        Bilaterate(x, Ancestors(0), Ancestors(1))
        for (i = 2; i < Ancestors.length; i++)
            UpdateBilateration(x, Ancestors(i))

Order(Node u, Node v, Node w)
    List Order = new List
    Order.add(u, v, w)
    List Shell = new List
    do
        foreach (Vertex x with at least 2 neighbors in Order)
            Shell.add(x)
        foreach (Vertex x in Shell)
            Order.add(x)
    while (Shell != null)
    return Order

Bilaterate(Node u, Node v, Node w)
    P_u = new List
    foreach (Position p_v of Node v)
        foreach (Position p_w of Node w)
            if (Consistent(p_v, p_w))
                [p_u1, p_u2] = CircleIntersection(p_v, p_w, d_uv, d_uw)
                p_u1.AncestorsByDepth = MergeAncestors(p_u, p_v)
                p_u2.AncestorsByDepth = p_w1.AncestorsByDepth
                P_u.add(p_u1, p_u2)
    u.setPositions(P_u)

UpdateBilateration(Node u, Node v)
    foreach (Position p_u of Node u)
        foreach (Position p_v of Node v)
            if (Consistent(p_u, p_v))
                d_uv = Measured distance between u and v
                if (p_u.distanceTo(p_v) == d_uv)
                    p_u.valid = true
                    p_v.valid = true
                    p_u.AncestorsByDepth = MergeAncestors(p_u, p_v)
    foreach (Position p_u of Node u)
        if (p_u.valid == false)
            P_u.remove(p_u)
    foreach (Position p_v of Node v)
        if (p_v.valid == false)
            P_v.remove(p_u)

Consistent(Position p_u, Position p_v)
    Position[] Ancestors_u = p_u.AncestorsByDepth
    Position[] Ancestors_v = p_v.AncestorsByDepth
    for (i = 0; i < max(Ancest_u.length, Ancest_v.length); i++)
        Ancest_u = Ancestors_u[i]
        Ancest_v = Ancestors_v[i]
        if (Ancest_u != null and Ancest_v != null)
            if (Ancest_u != Ancest_v)
                return false
    return true
```

Figure 4: The shell Sweeps localization algorithm.

eration graphs. We have never seen the graph in part (a) of the figure arise in practice, and the union of two globally rigid subnetworks connected by a non-bilaterable "bridge" as in part (b) is uncommon. We will exhibit in Section 5 that many globally rigid networks, especially globally rigid unit disk networks, are also in fact bilateration networks. This is significant because *globally rigid bilateration networks are uniquely localized by Sweeps*.



(a) The bipartite graph $K_{3,4}$ (b) Two copies of K_5

Figure 5: Graphs which are globally rigid but not bilateration graphs.

3.2 Sweeps with Distance and Angle Measurements

The Sweeps algorithm is very well suited to incorporate angle measurements in addition to distance information. An angle between two edges along with their lengths determines the distance between their distinct endpoints. Therefore, measuring the angle between every two distance measurements incident on a node is equivalent to "doubling" the network, *i.e.*, adding distance measurements between all nodes within two hops of each other.

Doubling a connected network gives a bilateration network [3, 14, 2], so this means that the Sweeps algorithm will succeed in finitely localizing a connected network with angle information. Furthermore, doubling a 2-edge connected network gives a globally rigid network, so it follows that doubling a 2-edge connected network gives a globally rigid bilateration network that Sweeps will uniquely localize.

As 2-edge connectivity is a mild condition on the connectivity of a network relative to the high density requirements of trilateration-based localization, this application of Sweeps could be very useful. An example scenario is an urban setting, where sensors could be deployed along streets in a minimally 2-edge connected fashion and still localize.

3.3 Sweeps with Noisy Distance Measurements

The Sweeps algorithm can be extended to handle noisy measurements. As noted in previous studies (*e.g.*, [28]), the larger the noise present in distance measurements, the more likely to occur are degenerate cases in which distance measurements to three nodes do not uniquely determine a node's position. This means that the basic Sweeps algorithm cannot succeed in always choosing the correct flip configuration. Because of this, we cannot eliminate potential node positions as we did before, or else we risk eliminating the correct configuration. We found that elimination criteria which use constraints on the position of only a single node result in correct positions being discarded with high frequency.

What we do instead is to consider groups of nodes together when deciding among flip configurations. Not all groups however have uniquely determined positions. Thus, we apply rigidity theory and identify groups of uniquely localizable nodes by identifying globally rigid components.

Specifically, we extend the shell Sweeps algorithm to partition nodes in each shell into two groups: those which are uniquely localizable when combined with constraints to nodes in previous shells, and those which are not. After nodes in a shell are finitely localized, all consistent combinations of positions of uniquely localizable nodes are produced. For each of these configurations, we calculate the squared discrepancy between the induced inter-node distances and the measured distances, which measures the *stress* of the configuration, $\sum_{(i,j)\in E_{gr}}(|x_i - x_j| - \hat{d}_{ij})^2$, where E_{gr} is the set of edges in the globally rigid component, x_i and x_j are computed possible positions for nodes i and j, and $\hat{d}_{ij}$ is the noisy measured distance between them. We use a simple approach in which we first eliminate the highest-stress configurations, and then choose the remaining configuration which violates the fewest unit-disk graph constraints. A different criterion could be used, but we have found this condition to work best among those tested. This approach works because each stage (shell) of the shell Sweep typically expands radially outward from previous shells. In general, an incorrectly flipped point tends to lie "inside" an earlier shell, at a position where it would have edges which it in fact does not have to previously swept nodes, while the correct position lies "outside", where it is less likely to violate unit-disk graph constraints.

The complete algorithm handling noisy measurements is outlined in Figure 6.

```
GloballyRigidShellSweep(Node u, Node v, Node w)
    List Order = Order(u, v, w)
    for (shellNum = 0; shellNum < numShells; shellNum++)
        foreach (Vertex x in Order and in Shell[shellNum])
            List Ancestors = (Neighbors of x earlier in Order)
            Bilaterate(x, Ancestors(0), Ancestors(1))
        FindGloballyRigidComponents(⋃_{i≤shellNum} Shell[i])
        foreach (GRComponent)
            Eliminate high stress configurations
            Choose config. violating fewest unit-disk graph constraints

FindGloballyRigidComponents(Nodes)
    if(induced graph not triconnected) then
      recurse on each triconnected component
    else if (not redundantly rigid) then
      recurse on each redundantly rigid component
    else return Nodes
```

Figure 6: Globally rigid shell Sweeps localization algorithm for noisy distance measurements.

4. ANALYSIS OF SWEEPS

Let $\mathbb{N}$ be a localizable network in the plane with nodes labeled $1, \ldots, n$. Let $\pi(1), \ldots, \pi(n)$ denote the positions of nodes $1, \ldots, n$ respectively. Suppose the set of node positions is generic, and more specifically, no three nodes are collinear. Let $\mathbb{G} = (\mathcal{V}, \mathcal{E})$ be the graph of $\mathbb{N}$, and assume $\mathbb{G}$ has a bilateration ordering. In the following we will show how the Sweeps algorithm can compute a position for each node such that all known inter-node distances are satisfied. The actual node positions can then be obtained by a Euclidean transformation using anchor positions.

For $v \in \mathcal{V}$, let $\mathcal{N}(v)$ denote the set of all nodes adjacent to v in $\mathbb{G}$. The Sweeps algorithm begins by selecting a bilateration ordering $[v] = v_1, \ldots, v_n$ of $\mathbb{G}$. Let $\mathcal{S}$ denote the set of seed nodes of $[v]$: $\mathcal{S} = \{v_1, v_2, v_3\}$. An ordering of all of the nodes of $\mathbb{G}$ is called a *sweep* of $\mathbb{N}$ just in case all of the nodes in $\mathcal{S}$ precede all of the other nodes, and there is at least one node in $\mathcal{V} - \mathcal{S}$ that is adjacent to a node preceding it in the ordering. An *assignment* is any function $\alpha : \mathcal{U} \to \mathbb{R}^2$ where $\mathcal{U} \subset \mathcal{V}$. Let $\mathcal{D}(\alpha)$ denote the domain of the assignment α. Two assignments α and α' are

said to be *consistent with each other* if $\alpha(u) = \alpha'(u)$ for all $u \in \mathcal{D}(\alpha) \cap \mathcal{D}(\alpha')$. Note that α and α' are consistent with each other if $\mathcal{D}(\alpha) \cap \mathcal{D}(\alpha') = \emptyset$. For $p \in \mathbb{R}^2$ and a positive real number r, let $\mathcal{C}(p, r)$ denote the circle of radius r centered at p.

The Sweeps algorithm selects the first sweep to be the bilateration ordering $[v]$ and *computes* the first sweep by computing a set $\mathcal{S}(v_i, 1)$ for each $i \in \{1, \ldots, n\}$ as follows. Assign a position to each of the seed nodes v_1, v_2, v_3 so that the known inter-node distances among them are satisfied. For seed node $v_i, i \in \{1, 2, 3\}$, let p_{v_i} denote the position assigned to v_i; define $\alpha_{p_{v_i}}$ to be the assignment with domain $\{v_i\}$ where $\alpha_{p_{v_i}}(v_i) = p_{v_i}$; define $\mathcal{S}(v_i, 1) = \{(p_{v_i}, \alpha_{p_{v_i}})\}$.

The Sweeps algorithm recursively computes the sets $\mathcal{S}(v_i, 1)$, $i > 3$, as follows. For $v_i, i > 3$, let $\mathcal{M}(v_i) = \mathcal{N}(v_i) \cap \{v_1, \ldots, v_{i-1}\}$. Since $[v]$ is a bilateration ordering, $\mathcal{M}(v_i)$, $i > 3$, must be a set of at least two elements. Let $u_1, \ldots, u_m$ be the elements of $\mathcal{M}(v_i)$. Let $\mathcal{S}(v_i, 1)$ be the set of all (p, α_p) computed as follows:

1. There exist $(p_{u_j}, \alpha_{p_{u_j}}) \in \mathcal{S}(u_j, 1)$, $j \in \{1, \ldots, m\}$, such that $\alpha_{p_{u_j}}$ is consistent with $\alpha_{p_{u_h}}$ for all $j, h \in \{1, \ldots, m\}$, and $p \in \bigcap_{j \in \{1,\ldots,m\}} \mathcal{C}(p_{u_j}, d_{u_j v_i})$.

2. Let $\alpha_p(v_i) = p$. For $w \in \bigcup_{h \in \{1,\ldots,m\}} \mathcal{D}(\alpha_{p_{u_h}})$, we let $\alpha_p(w) = \alpha_{p_{u_j}}(w)$ where $w \in \mathcal{D}(\alpha_{p_{u_j}})$ for some $j \in \{1, \ldots, m\}$.

Note that α_p is well defined because $\alpha_{p_{u_j}}$ is consistent with $\alpha_{p_{u_h}}$ for all $j, h \in \{1, \ldots, m\}$, and only $\mathcal{S}(u_j, 1), j \in \{1, \ldots, m\}$, are used in computing $\mathcal{S}(v_i, 1)$. It is straightforward to show that $\mathcal{S}(u, 1)$ is non-empty and consists of a finite number of elements for each $u \in \mathcal{V}$. We say Sweeps has *computed the first sweep* when all $\mathcal{S}(v_i, 1), i \in \{1, \ldots, n\}$, are computed.

Sweeps computes the kth sweep, $k > 1$, by selecting a sweep $[u]$ distinct from the previous $k - 1$ sweeps, and computing a set $\mathcal{S}(u_i, k)$ for each $i \in \{1, \ldots, n\}$ as follows. Note that $[u]$ does not have to be a bilateration ordering if it is not the first sweep. Let $\mathcal{M}(u_i) = \mathcal{N}(u_i) \cap \{u_1, \ldots, u_{i-1}\}$. Let $\mathcal{S}(u_i, k) = \mathcal{S}(u_i, k - 1)$ if $i \in \{1, 2, 3\}$ or $\mathcal{M}(u_i) = \emptyset$.

Consider $u_i, i > 3$, where $\mathcal{M}(u_i)$ is non-empty. Let $w_1, \ldots, w_m$ be the elements of $\mathcal{M}(u_i)$. Let $\mathcal{S}(u_i, k)$ be the set of all (p, α_p) computed as follows:

1. There exist $(p, \alpha'_p) \in \mathcal{S}(u_i, k - 1)$ for some α'_p, and also $(p_{w_j}, \alpha_{p_{w_j}}) \in \mathcal{S}(w_j, k)$, $j \in \{1, \ldots, m\}$, such that the assignments α'_p and $\alpha_{p_{w_j}}$, $j \in \{1, \ldots, m\}$, are pairwise consistent, and $p \in \bigcap_{j \in \{1,\ldots,m\}} \mathcal{C}(p_{w_j}, d_{u_i w_j})$.

2. Define α_p as follows: For $x \in \mathcal{D}(\alpha'_p) \cup \bigcup_{h \in \{1,\ldots,m\}} \mathcal{D}(\alpha_{p_{w_h}})$, let $\alpha_p(x) = \alpha'_p(x)$ if $x \in \mathcal{D}(\alpha'_p)$, and let $\alpha_p(x) = \alpha_{p_{w_j}}(x)$ if $x \in \mathcal{D}(\alpha_{p_{w_j}})$ for some $j \in \{1, \ldots, m\}$.

Note that α_p is well defined because the assignments α'_p and $\alpha_{p_{w_j}}, j \in \{1, \ldots, m\}$, are pairwise consistent, and only $\mathcal{S}(u_i, k-1)$ and $\mathcal{S}(w_j, k), j \in \{1, \ldots, m\}$, are used in computing $\mathcal{S}(u_i, k)$. It is straightforward to show that $\mathcal{S}(w, k)$ is non-empty and consists of a finite number of elements for each $w \in \mathcal{V}$. We say Sweeps has *computed the k-th sweep* when all $\mathcal{S}(u_i, k)$, $i \in \{1, \ldots, n\}$, are computed.

A *path* from node a to b in $\mathbb{G}$ is a sequence of nodes $a_1, \ldots, a_l$ where a is adjacent to a_1, a_i is adjacent to $a_{i+1}, i \in \{1, \ldots, l-1\}$, and a_l is adjacent to b. Let $[w] = w_1, \ldots, w_n$ be a sweep. For $w_j \in \mathcal{V} - \mathcal{S}$, define $\mathbb{G}(w_j, [w])$ as the subgraph of $\mathbb{G}$ induced by w_j and all nodes $w_i, i < j$, where w_i is adjacent to w_j or there exists a path $w_{i_1}, \ldots, w_{i_m}$ from w_i to w_j such that $i < i_k < i_{k+1} < j$ for $k \in \{1, \ldots, m-1\}$. If node w_j is in $\mathbb{G}(w_i, [w])$, then we write $w_j \in \mathbb{G}(w_i, [w])$.

LEMMA 1. *Suppose Sweeps has computed k sweeps where $k \geq 1$, and let $[w]$ be the kth sweep. Suppose $(p_u, \alpha_{p_u}) \in \mathcal{S}(u, k)$ where $u \in \mathcal{V} - \mathcal{S}$.*

1. *If node $v \in \mathbb{G}(u, [w])$, then $v \in \mathcal{D}(\alpha_{p_u})$.*
2. *If nodes a and b are adjacent in $\mathbb{G}(u, [w])$, then $||\alpha_{p_u}(a) - \alpha_{p_u}(b)|| = d_{ab}$.*
3. *If $v \in \mathcal{S}$ and $v \in \mathcal{D}(\alpha_{p_u})$, then $\alpha_{p_u}(v) = p_v$, where p_v is the position assigned to v.*

Proof: See Appendix. □

Consider $u \in \mathcal{V} - \mathcal{S}$. Let $\mathcal{N}_1(u)$ denote the set of nodes in $\mathcal{V}$ adjacent to u. Suppose for some integer $i \geq 1$, $\mathcal{N}_j(u)$, $j \in \{1, \ldots, i\}$ have been determined. Let $\mathcal{N}_{i+1}(u)$ denote the set of nodes $w \in \mathcal{V}$ where $w \notin \bigcup_{j \in \{1,\ldots,i\}} \mathcal{N}_j(u)$ and w is adjacent to a node in $\mathcal{N}_i(u) - \mathcal{S}$. Since there are a finite number of nodes, there can be only a finite number of sets generated this way. Suppose we have h sets generated this way: $\mathcal{N}_1(u), \ldots, \mathcal{N}_h(u)$. We call $\mathcal{N}_i(u)$, $i \in \{1, \ldots, h\}$, the *path sets* of node u. Let $u_n = |\bigcup_{i \in \{1,\ldots,h\}} \mathcal{N}_i(u) - \mathcal{S}| + 1$. Select any u_n elements of $\{4, \ldots, n\}$ and label them as $i_1, i_2, \ldots, i_{u_n}$ so that $i_1 < i_2 < \ldots < i_{u_n}$. We construct a "complete sweep" for u as follows. Assign indices 1 to 3 to nodes in $\mathcal{S}$ in any manner, and assign index i_{u_n} to node u. Assign indices $i_{u_n - 1}$ to $i_{u_n - 1 - |\mathcal{N}_1(u) - \mathcal{S}|}$ to the nodes in $\mathcal{N}_1(u) - \mathcal{S}$. Similarly, assign indices $i_{u_n - 1 - |\mathcal{N}_1(u) - \mathcal{S}|) - 1}$ to $i_{u_n - 1 - |\mathcal{N}_1(u) - \mathcal{S}|) - 1 - |\mathcal{N}_2(u) - \mathcal{S}|}$ to the nodes in $\mathcal{N}_2(u) - \mathcal{S}$. And so on. Assign the indices in $\{4, \ldots, n\} - \{i_1, \ldots, i_{u_n}\}$ to the remaining nodes in any manner. The resulting ordering $c_1, \ldots, c_n$ is a sweep since the nodes in $\mathcal{S}$ precede all other nodes, and node u is adjacent to at least one node preceding it. We call this ordering a *complete sweep* for u.

LEMMA 2. *Let $u \in \mathcal{V} - \mathcal{S}$, and suppose u has path sets $\mathcal{N}_1(u)$, $\mathcal{N}_2(u), \ldots, \mathcal{N}_h(u)$.*

1. *The set $\bigcup_{i \in \{1,\ldots,h\}} \mathcal{N}_i(u)$ is equal to the set of all nodes $x \in \mathcal{V}$ that is either adjacent to u or has a path $x_1, \ldots, x_m$ to u, where $x_i \in \mathcal{V} - \mathcal{S}$, $i \in \{1, \ldots, m\}$.*
2. *Suppose $[c]$ is a complete sweep for node u. Then $\mathbb{G}(u, [c])$ is the subgraph of $\mathbb{G}$ induced by u and all nodes $x \in \mathcal{V}$ that is either adjacent to u or has a path $x_1, \ldots, x_m$ to u, where $x_i \in \mathcal{V} - \mathcal{S}$, $i \in \{1, \ldots, m\}$.*

Proof: See Appendix. □

Consider the subgraph $\mathbb{H}$ of $\mathbb{G}$ induced by nodes in $\mathcal{V} - \mathcal{S}$. Let $\mathbb{H}_1, \ldots, \mathbb{H}_m$ be the maximally connected components of $\mathbb{H}$. Let u_i be a node of $\mathbb{H}_i$ for each $i \in \{1, \ldots, m\}$. Construct a complete sweep for u_1, then use the left over indices to construct a complete sweep for u_2, and so on. We call the resulting sweep a *complete sweep of the network*, and we say that the sweep is *based on* $u_1, \ldots, u_m$. Lemmas 1 and 2 can be used to show the following:

LEMMA 3. *Select two sweeps, the first of which is the bilateration ordering $[v]$ of $\mathbb{G}$, and the second of which is a complete sweep of the network. Let $u_1, \ldots, u_m$ be the nodes on which the complete sweep is based.*

1. *$\mathcal{S}(u_i, 2)$ is a singleton for all $i = 1, \ldots, m$.*

2. *Suppose $\mathcal{S}(u_i, 2) = \{(p_i, \alpha_{p_i})\}$ for $i \in \{1, \ldots, m\}$. Define the assignment $\alpha : \mathcal{V} \to \mathbb{R}^2$ as $\alpha(w) = \alpha_{p_i}(w)$ if $w \in \mathcal{D}(\alpha_{p_i})$. Then α is well defined, and $||\alpha(w) - \alpha(x)|| = d_{wx}$ for all adjacent nodes w and x in $\mathbb{G}$.*

A direct consequence of Lemma 3 is the following:

THEOREM 1. *A localizable bilateration network $\mathbb{N}$ can be localized by the Sweeps algorithm in two sweeps, the first of which is a bilateration ordering, and the second of which is a complete sweep of the network, followed by an Euclidean transformation using anchor positions.*

5. EVALUATIONS

5.1 Precise Distance Measurements

We first evaluate the performance of Sweeps with precise distance measurements. We generate uniformly random networks of 250 nodes in a square region. We do not consider anchors, as we are interested here in how many nodes Sweeps can localize. In all our measurements, we aggregate the results from 100 network instances at each sensing radius and compute 95th-percentile confidence intervals for each quantity of interest.

We use sensing range to control the density and connectivity of the network. In random networks, there is a sensing radius r which ensures k-connectivity as well as trilaterability with high probability [3]. This is important because 3-connectivity is a necessary condition for unique localizability. The results of this section all stem from the fact that an average degree will ensure bilaterability with a certain (possibly low) probability, but this average degree is *lower* than that required for trilaterability with the same probability.

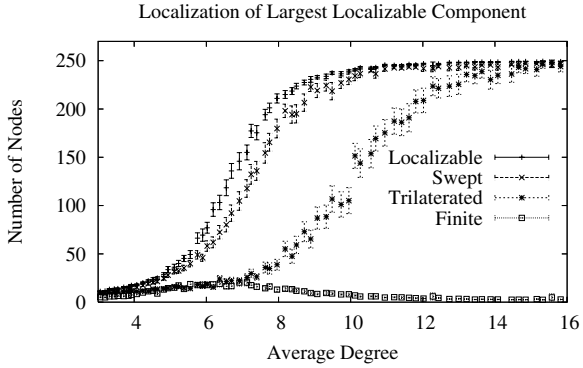


Figure 7: Numbers of nodes localizable, Sweeps localized, trilateration localized, and finitely localized vs. average network connectivity in random networks.

Figure 7 reports our results. To generate this figure, we first identify all theoretically uniquely localizable nodes by finding the globally rigid components. Then we start a sweep at a random edge in the largest globally rigid component. The number of uniquely localizable nodes swept by our algorithm is labeled as Swept. From this figure, it is clear that the number of swept nodes is very close to the number of uniquely localizable nodes. Our algorithm also identifies nodes with finite but not uniquely determined positions. These nodes are labeled as Finite. For comparison, this figure also plots the number of nodes whose positions are determined by iterated trilateration. We observe that our algorithm significantly outperforms trilateration.

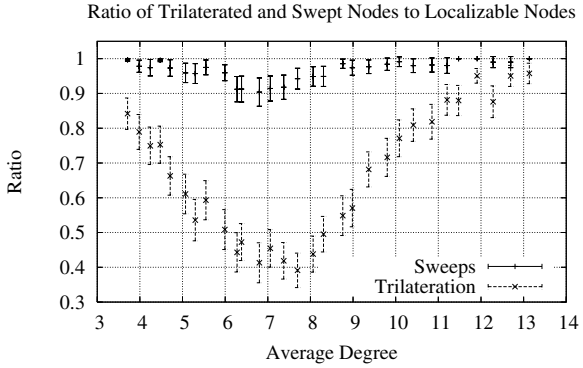


Figure 8: Ratios of the number of nodes localized to the number of nodes that is theoretically possible to localize.

The performance gap between our algorithm and trilateration is highlighted in Figure 8. We find that iterated trilateration always localizes fewer nodes than a sweep started at the same nodes because a trilateration network is necessarily a bilateration network but the converse is not true. Specifically, *Sweeps localizes at least 90% of uniquely localizable nodes with high confidence in random networks when the network density varies in a wide range*. For comparison, trilateration guarantees localization of only 40% of nodes. Most sequential algorithms operate only on trilateration graphs [28], so to the best of our knowledge, no such algorithm succeeds in localizing more nodes than Sweeps.

Figure 8 also illustrates the transition from non-localizable to localizable and sweepable networks. At around average degree 6 to 8, Sweeps localizes a lower proportion of localizable nodes than it does elsewhere. It was observed and theoretically justified in [17] that this is the connectivity level at which networks start making the transition from containing many small globally rigid components to a single large component. At this density, large globally rigid components start to absorb small peripheral components, and non-bilaterable structures appear at the edges of the globally rigid component. Even in this problematic regime, the extent of localization remains around 90%.

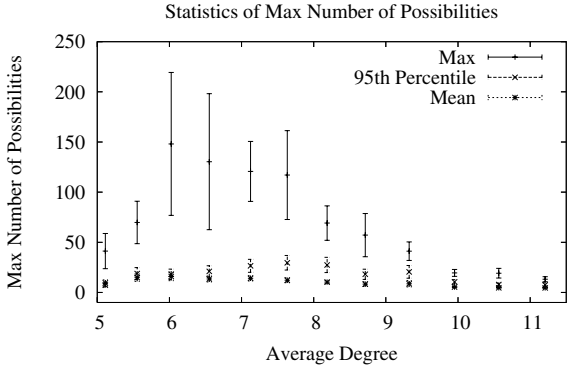


Figure 9: Over each run of Sweeps localization, the maximum number of possibilities kept at any time for a single node, the average number of possibilities kept, and the 95th-percentile number of possibilities kept.

As worst-case exponential computational complexity is a potential concern, we test the running time of Sweeps. We found that potential exponential growth in possibilities rarely develops to a point where the algorithm fails to complete in minutes on a consumer PC even for networks of a thousand nodes. At a given density, typically, the running time of Sweeps grows linearly with increasing number of nodes. It localizes more than 95% of uniquely localiz-

able nodes in a network of one-thousand nodes with average degree less than five in two minutes on a consumer PC with an Intel Xeon CPU of 2.80 GHz. Thus, a potentially more serious problem is state keeping. We run Sweeps and keep track of the maximum number of possibilities kept at any time for any node over the course of the algorithm, the mean number of possibilities kept for all nodes, and the 95th percentile number of possibilities. These numbers are presented in Figure 9. We observe that on these 250-node networks, the total number of possibilities maintained by the algorithm per node is less than 50 with high confidence. While we will not show the results here, we also tracked the average number of possibilities kept per node as network size increased while holding average degree constant at 6, and found that this quantity remained constant over the entire range tested.

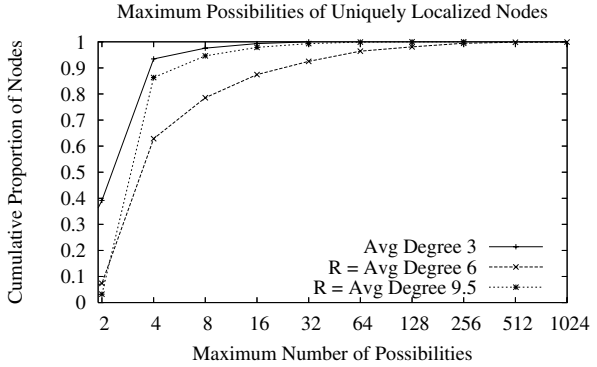


Figure 10: Histogram of maximum number of possible positions of each node kept by the Sweeps algorithm at different average network connectivity.

Intermediate densities around the above-mentioned transition between many small globally rigid clusters and a single globally rigid component again prove the most challenging for the algorithm. At this density, even though we sweep on a uniquely localizable component, there are sometimes a few shells over which several bilaterations take place consecutively without any redundant edges to eliminate possibilities. In Figure 10, we see further verification of the efficiency of Sweeps. For the most difficult density of around average degree 6, 90% of nodes never have more than 8 possibilities. Less than 0.5% ever have more than 256 possible positions.

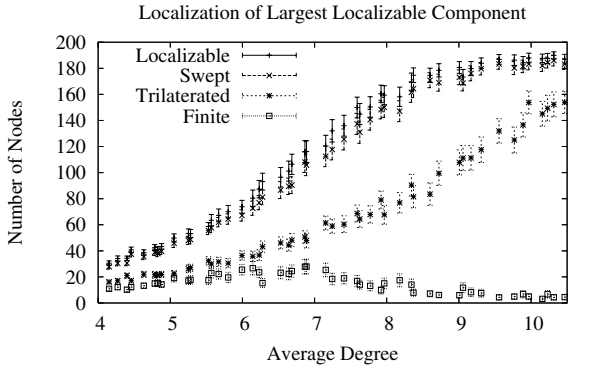


Figure 11: Numbers of nodes localizable, Sweeps localized, trilateration localized, and finitely localized vs. average network connectivity in random networks deployed around an opaque obstacle.

We also run our simulations on anisotropic networks in which 250 nodes are randomly deployed in a ring around a large opaque rectangular obstacle which occupies one-half of the deployment

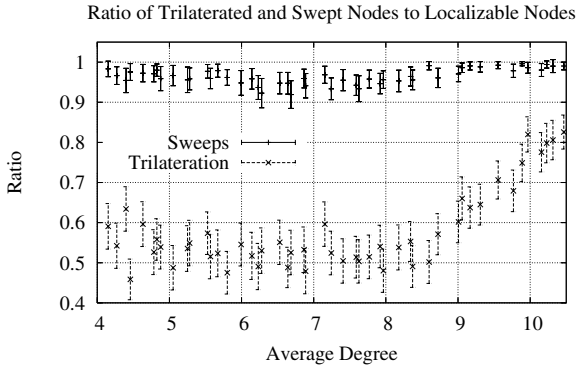


Figure 12: Ratios of the number of nodes localized to the number of nodes that is theoretically possible to localize. Even in the presence of a large obstacle, Sweeps consistently localizes a high percentage of localizable nodes.

area. The results on the extent of Sweeps localization and its complexity are very much similar to the random case. Some of these results are shown in Figures 11 and 12.

5.2 Sweeps in Regularly Deployed Networks

In this section, we consider localization on regularly deployed networks. Random deployment can have unpredictable spatial coverage and localizability due to local non-uniformity in node placement.

In order to study localization on regular networks, we implement a mobility control rule that allows us to eliminate empty regions in the network and achieve a target average node degree. We initially deploy nodes randomly at a high density and then evolve the network to achieve the target average node degree uniformly. The details of the mobility scheme are specified in Appendix B. The standard deviation in node degree after mobility control is consistently less than half of that of a network randomly deployed across the same region.

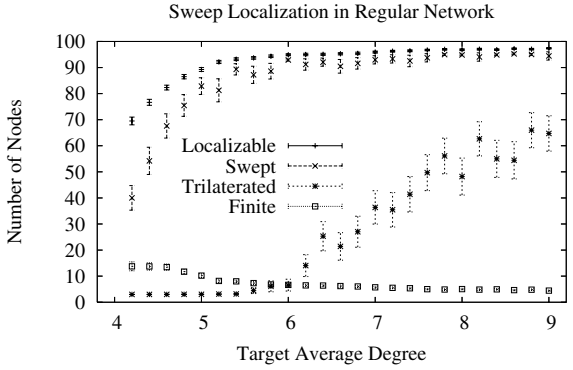


Figure 13: Numbers of nodes localizable, Sweeps localized, trilateration localized, and finitely localized vs. average network connectivity in regular networks.

One might think that regular networks would be straightforward to localize, but this turns out not to be the case. For instance, at low density, the near-symmetries which appear can make localization more difficult for global optimization approaches by making local minima more likely. On uniquely localizable unions of wheel networks (a honeycomb pattern), SDP localization often outputs a flip configuration even with no noise present in the distance measurements. As we saw before and will soon see again, sequential approaches based on trilateration networks are also likely ineffective

on sparse regularly deployed networks, especially wheel networks. We see in Figure 13 that at varying regular network density, the proportion of localizable nodes which are trilaterable is very low even at high density, while Sweeps is again effective.

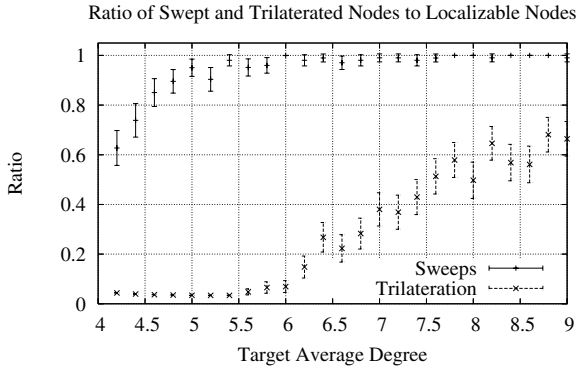


Figure 14: Ratios of the number of nodes localized to the number of nodes that is theoretically possible to localize in regular networks. Sweeps performs particularly well on these networks.

The performance gap between our algorithm and that of trilateration on regular networks is highlighted in Figure 14. We observe that Sweeps is still effective, while the performance of trilateration is extremely poor. If we compare these results with those for random networks in Figure 8, we can see that Sweeps is more effective in regular networks, while trilateration-based approaches and global optimization are less effective in this setting.

5.3 Sweeps with Noisy Distance Measurements

Finally, we evaluate the performance of the version of our Sweeps algorithm adapted for noisy distance measurements. We add zero-mean Gaussian noise with standard deviation of 1, 5, and 15% of the sensing range to all distance measurements. [2] The number of shells upon which our algorithm successfully localizes nodes depends on the magnitude of the distance errors. We have found that the localization out to five shells is robust to flip configurations at the noise levels tested.

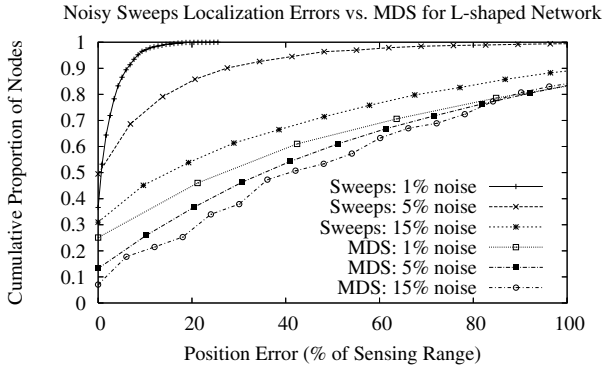


Figure 15: Cumulative proportion of nodes with less than given localization error for a random deployment of 50 nodes over an L-shaped region.

In evaluation on those networks upon which global optimization approaches tend to do poorly, we find that Sweeps remains effective. On non-convex deployments where MDS struggles, Sweeps computes good position estimates, as shown in Figure 15. Recent improvements to MDS [24] improve the estimation of the complete distance matrix, but do so by bootstrapping from anchor positions. Sweeps requires no anchors, and we have simulated the anchor-free case. On regular deployments where SDP usually produces a flip configuration, Sweeps also succeeds, as shown in Figure 16.

[2] We do not consider the case in which there may be severe outliers in distance measurements. This is a challenging problem that has begun to be investigated by Berger *et al.* [6].

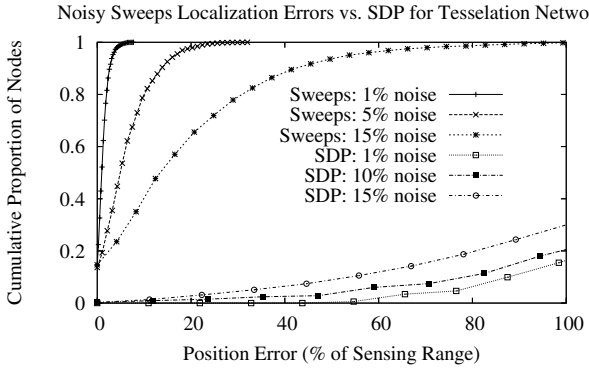


Figure 16: Cumulative proportion of nodes with less than given localization error for a 26-node uniquely localizable union of wheel networks.

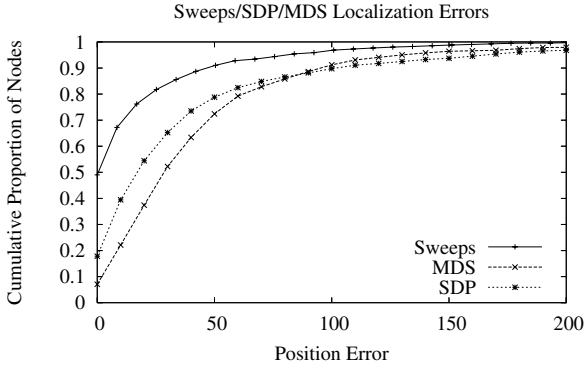


Figure 17: Cumulative proportion of nodes with less than given localization error using Sweeps, MDS, and SDP.

Finally, we simulate Sweeps on random networks of 100 nodes and 5 anchors at an average degree of 8, with 5% Gaussian noise in all distance measurements. We then start a sweep at each anchor and limit the depth of the sweep to five in order to avoid the possibility of large flip errors. We find this approach effective, as localized nodes have lower estimation errors on average than nodes localized using MDS [37] or SDP [7], as shown in Figure 17. This is partly due to the fact that Sweeps localizes only localizable nodes and avoids large errors due to unrecognized flip configurations.

6. RELATED WORK

Network localization is an active research field (*e.g.*, [5, 9, 10, 18, 19, 20, 23, 24, 25, 33, 38]). The previous approaches can be roughly classified into two types. The first type is called coarse-grained or range free localization. The focus of this paper is on the second type—fine-grained localization. Thus, we review only previous work on fine-grained localization (*e.g.*, [1, 7, 8, 11, 13, 21, 28, 29, 30, 31, 32, 34, 35, 36, 39]). Eren *et al.* studied the theoretical conditions for fine-grained localization in [14, 3, 17]. These conditions are applied in various settings. For instance, in [32], the authors proposed an algorithm using mobility to obtain distance measurements which result in globally rigid constraint structures.

Many fine-grained localization algorithms are based on global optimization. In particular, Biswas *et al.* applied semidefinite programming (SDP) to fine-grained localization [8, 7]. Their algorithms are effective in relatively dense over-constrained networks. Specifically, their algorithms require that $\Omega(n^2)$ pairs of nodes know their relative distances, where n is the number of sensor nodes in the network. In sparse networks or networks with sparse subregions, their algorithms may not be able to correctly localize. An alternative to SDP is multidimensional scaling (MDS) (*e.g.*, [21, 25, 36]). As we discussed in Section 1, MDS requires an initial estimation of the complete distance matrix, which may be available only in dense networks. Neither SDP or MDS can identify all positions.

The Sweeps algorithm belongs to the type of sequential localization algorithms. Other sequential localization algorithms have been proposed before (*e.g.*, [28, 35]). In particular, in [28], Moore *et al.* proposed a sequential localization algorithm based on trilateration graphs under noisy distance measurements. However, their algorithm is effective only in relatively dense networks, while our Sweeps algorithm localizes a much larger class of network.

In prior work [15, 16], Fang *et al.* first proposed the idea of sweeping through a bilateration network in a sequential fashion. In their algorithm, possibly non-consistent combinations of node positions were used to compute further possibilities. In this paper we improve computational complexity using consistent position combinations and shell sweeps, extend the idea to handle angle and noisy measurements, and provide extensive evaluations.

7. CONCLUSION AND FUTURE WORK

Our work succeeds in provably localizing sparser networks. One reason we believe this is an important contribution is that it extends, in practice, the class of networks for which feasible localization algorithms are known to those with little more than the minimum number of constraints necessary for any algorithm to succeed. Since Sweeps is an incremental approach, it will be amenable to a distributed implementation, but we are leaving this for future work.

We have also shown that our algorithm exhibits a synergy with a scheme for coverage-optimizing controlled mobility, resulting in a promising unified design for simultaneous spatial coverage, localizability optimization, and localization. We envision joint controlled mobility-localization to be an eminently practical and effective network model with which to circumvent the inherent NP-hardness of localization by altering network connectivity through mobility so as to be efficiently localizable by a particular algorithm.

8. ACKNOWLEDGMENTS

Goldenberg and Yang was supported in part, by grants from the U.S. NSF. Fang, Cao, and Morse was supported in part, by grants from the U.S. Army Research Office and the U.S. NSF and by a gift from the Xerox Corporation. Bihler was supported by a grant from the German National Academic Foundation and the U.S. NSF. Anderson was supported by an Australian Research Council Discovery Projects Grant and by National ICT Australia, which is funded by the Australian Government's Department of Communications, Information Technology and the Arts and Australian Research Council through the Backing Australia's Ability initiative and the ICT Centre of Excellence Program. We are grateful to Jennifer Hou, our shepherd, for her help in revising the paper. We are also grateful to the anonymous reviewers whose comments improve the paper.

9. REFERENCES

[1] J. Albowicz, A. Chen, and L. Zhang. Recursive position estimation in sensor networks. In *Proceedings of the 9th International Conference on Network Protocols (ICNP)*, pages 35–41, Riverside, CA, Nov. 2001.

[2] B. Anderson, P. Belhumeur, T. Eren, D. Goldenberg, A. Morse, W. Whiteley, and Y. R. Yang. Global properties of easily localizable sensor networks. Preprint Australian National University, 2005.

[3] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 2006.

[4] J. Aspnes, D. Goldenberg, and Y. R. Yang. On the computational complexity of sensor network localization. In *Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, Turku, Finland, July 2004.

[5] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.

[6] B. Berger, J. Kleinberg, and T. Leighton. Reconstructing a three-dimensional model with arbitrary errors. *Journal of the ACM (JACM)*, 46(2):212–235, 1999.

[7] P. Biswas, T.-C. Liang, K.-C. Toh, T.-C. Wang, and Y. Ye. Semidefinite programming approaches to sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*, 2006.

[8] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the Third International Workshop on Information Processing in Sensor Networks (IPSN'04)*, Berkeley, CA, Apr. 2004.

[9] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, Oct. 2000.

[10] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proceedings of HICSS*, 2001.

[11] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal. Ad-hoc localization using ranging and sectoring. In *Proceedings of IEEE INFOCOM*, Hong Kong, Apr. 2004.

[12] J. Cortes and F. Bullo. Coordination and geometric optimization via distributed dynamical systems. *SIAM Journal on Control and Optimization*, 44:1543–1574, 2005.

[13] L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[14] T. Eren, D. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur. Rigidity, computation, and randomization in network localization. In *Proceedings of IEEE INFOCOM*, Hong Kong, Apr. 2004.

[15] J. Fang, M. Cao, A. S. Morse, and B. D. O. Anderson. Localization of sensor networks using Sweeps. In *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006.

[16] J. Fang, M. Cao, A. S. Morse, and B. D. O. Anderson. Sequential localization of networks. In *Proceedings of Seventeenth International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan, July 2006.

[17] D. Goldenberg, A. Krishnamurthy, W. Maness, Y. R. Yang, A. Young, A. S. Morse, A. Savvides, and B. D. O. Anderson. Network localization in partially localizable networks. In *Proceedings of IEEE INFOCOM*, Miami, FL, Apr. 2005.

[18] A. Haeberlen, E. Flannery, A. Ladd, A. Rudys, D. Wallach, and L. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the Tenth International Conference on Mobile Computing and Networking (Mobicom)*, Philadelphia, PA, Sept. 2004.

[19] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes in large scale sensor networks. In *Proceedings of the Ninth International Conference on Mobile Computing and Networking (Mobicom)*, pages 81–95, San Diego, CA, Sept. 2003.

[20] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proceedings of the Tenth International Conference on Mobile Computing and Networking (Mobicom)*, Philadelphia, PA, Sept. 2004.

[21] X. Ji and H. Zha. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. In *Proceedings of IEEE INFOCOM*, Hong Kong, Apr. 2004.

[22] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 2002.

[23] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43:499–518, 2003.

[24] H. Lim and J. Hou. Localization for anisotropic sensor networks. In *Proceedings of IEEE INFOCOM*, Miami, FL, Apr. 2005.

[25] H. Lim, L. Kung, J. Hou, and H. Luo. Zero-configuration, robust indoor localization: theory and experimentation. In *Proceedings of IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[26] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem - The asynchronous case. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004.

[27] J. McLurkin and J. Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Proceedings of Distributed Autonomous Robotic Systems Conference*, 2004.

[28] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, Nov. 2004.

[29] D. Niculescu and B. Nath. Ad-hoc positioning system. In *Proceedings of IEEE Globecom*, San Antonio, TX, Nov. 2001.

[30] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, Apr. 2003.

[31] D. Niculescu and B. Nath. VOR basestations for indoor 802.11 positioning. In *Proceedings of the Tenth International Conference on Mobile Computing and Networking (Mobicom)*, Philadelphia, PA, Sept. 2004.

[32] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, Miami, FL, Apr. 2005.

[33] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *Proceedings of the Sixth International Conference on Mobile Computing and Networking (Mobicom)*, pages 32–43, Boston, MA, Aug. 2000.

[34] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *Proceedings of USENIX Technical Annual Conference*, Monterey, CA, June 2002.

[35] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the Seventh International Conference on Mobile Computing and Networking (Mobicom)*, pages 166–179, Rome, Italy, July 2001.

[36] Y. Shang and W. Ruml. Improved MDS-based localization. In *Proceedings of IEEE INFOCOM*, Hong Kong, Apr. 2004.

[37] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *Proceedings of the Fourth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Annapolis, MD, June 2003.

[38] R. Stoleru, T. He, J. Stankovic, and D. Luebke. High-accuracy, low-cost localization system for wireless sensor network. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2005.

[39] C. Wang and L. Xiao. Locating sensors in concave environments. In *Proceedings of IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

APPENDIX

A. PROOFS OF LEMMA 1 AND LEMMA 2

Proof of Lemma 1:

1. Suppose $v \in \mathbb{G}(u,[w])$. If $v = u$ or v is adjacent to u, then it is a direct consequence of how α_{p_u} is computed in Sweeps that $u \in \mathcal{D}(\alpha_{p_u})$. So $v \in \mathcal{D}(\alpha_{p_u})$ when $v = u$.

   Suppose $v \neq u$. This means in the kth sweep $[w]$, we have a path $w_{i_1}, \ldots, w_{i_m}$ from v to u. Since $(p_u, \alpha_{p_u}) \in \mathcal{S}(u,k)$, this means there is a $(p_{w_{i_m}}, \alpha_{p_{w_{i_m}}}) \in \mathcal{S}(w_{i_m},k)$ such that $\alpha_{p_{w_{i_m}}}$ is consistent with α_{p_u} and $\mathcal{D}(\alpha_{p_{w_{i_m}}}) \subset \mathcal{D}(\alpha_{p_u})$. This is a direct consequence of how Sweeps computes $\mathcal{S}(u,k)$. Similarly, $(p_{w_{i_m}}, \alpha_{p_{w_{i_m}}}) \in \mathcal{S}(w_{i_m},k)$ implies there exists $(p_{w_{i_{m-1}}}, \alpha_{p_{w_{i_{m-1}}}}) \in \mathcal{S}(w_{i_{m-1}},k)$ such that $\alpha_{p_{w_{i_{m-1}}}}$ is consistent with $\alpha_{p_{w_{i_m}}}$ and $\mathcal{D}(\alpha_{p_{w_{i_{m-1}}}}) \subset \mathcal{D}(\alpha_{p_{i_{w_m}}}) \subset \mathcal{D}(\alpha_{p_u})$. By repeating this argument for $w_{i_{m-2}}, \ldots, w_{i_1}, v$, we get that there is $(p_v, \alpha_{p_v}) \in \mathcal{S}(v,k)$ such that $\mathcal{D}(\alpha_{p_v}) \subset \mathcal{D}(\alpha_{p_u})$. Since $v \in \mathcal{D}(\alpha_{p_v})$, we have that $v \in \mathcal{D}(\alpha_{p_u})$.

2. Suppose nodes a and b are adjacent nodes in $\mathbb{G}(u,[w])$. As shown above, $a, b \in \mathcal{D}(\alpha_{p_u})$. If $a, b \in \mathcal{S}$, then it is easy to see that $\parallel \alpha_{p_u}(a) - \alpha_{p_u}(b) \parallel = d_{ab}$. So suppose at least one of a or b is not in $\mathcal{S}$, and without loss of generality, suppose a precedes b in $[w]$. Hence, there is a path $w_{i_1} = b, w_{i_2}, \ldots, w_{i_m}$ from $a = w_{i_0}$ to $u = w_{i_{m+1}}$ where $i_0 < i_1 < \ldots < i_m < i_{m+1}$.

   For any $(p_b, \alpha_{p_b}) \in \mathcal{S}(b,k)$, it is straightforward to show that $a \in \mathcal{D}(\alpha_{p_b})$ and $\parallel \alpha_{p_b}(a) - \alpha_{p_b}(b) \parallel = d_{ab}$.

   Using the same logic as in part 1, we have that there exist $(p_{w_{i_1}}, \alpha_{p_{w_{i_1}}}) \in \mathcal{S}(w_{i_1},k)$, $(p_{w_{i_2}}, \alpha_{p_{w_{i_2}}}) \in \mathcal{S}(w_{i_2},k), \ldots,$ $(p_{w_{i_m}}, \alpha_{p_{w_{i_m}}}) \in \mathcal{S}(w_{i_m},k)$ such that $\alpha_{p_{w_{i_l}}}$ is consistent with $\alpha_{p_{w_{i_{l+1}}}}$ for all $l \in \{1, \ldots, m-1\}$, and $\alpha_{p_{w_{i_m}}}$ is consistent with α_{p_u}. Since $w_{i_1} = b$, we have that $\parallel \alpha_{p_{w_{i_1}}}(a) - \alpha_{p_{w_{i_1}}}(b) \parallel = d_{ab}$. This and part 1 imply $\parallel \alpha_{p_{w_{i_m}}}(a) - \alpha_{p_{w_{i_m}}}(b) \parallel = d_{ab}$. Hence, we get $\parallel \alpha_{p_u}(a) - \alpha_{p_u}(b) \parallel = d_{ab}$.

3. This is direct consequence of the Sweeps algorithm. □

Proof of Lemma 2:

1. Suppose $x \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. This implies $x \in \mathcal{N}_i(u)$ for some $i \in \{1,\ldots,h\}$. Hence x is either adjacent to u or to some $x_{i-1} \in \mathcal{N}_{i-1}(u) - \mathcal{S}$. Similarly x_{i-1} is either adjacent to u or to some $x_{i-2} \in \mathcal{N}_{i-2}(u) - \mathcal{S}$. And so on until we get $x_1 \in \mathcal{N}_1(u) - \mathcal{S}$. So either x is adjacent to u or there is a path $x_{i-1}, x_{i-2}, \ldots, x_1$ from x to u where $x_j \in \mathcal{V} - \mathcal{S}$ for all $j \in \{1,\ldots,i-1\}$.

   Suppose x is adjacent to u. Then $x \in \mathcal{N}_1(u)$. Now suppose $x \in \mathcal{V}$ has a path $x_m, x_{m-1}, \ldots, x_1$ to u where $x_j \in \mathcal{V} - \mathcal{S}$, for all $j \in \{1,\ldots,m\}$. By definition of path sets , $x_1 \in \mathcal{N}_1(u)$ since it is adjacent to u. Since $x_1 \in \mathcal{N}_1(u) - \mathcal{S}$, we have that $x_2 \in \bigcup_{i=1,2} \mathcal{N}_i(u)$ since x_1 and x_2 are adjacent. Suppose for some $g \in \{1,\ldots,m-1\}$ that all x_l, $l \le g$, are in $\bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. Suppose $x_g \in \mathcal{N}_{i_g}(u)$. Consider x_{g+1}. Since x_{g+1} is adjacent to x_g and $x_g \in \mathcal{N}_{i_g}(u) - \mathcal{S}$, we have that $x_{g+1} \in \mathcal{N}_{i_g+1}(u) \cup \mathcal{N}_{i_g} \ldots \cup \mathcal{N}_1$. If $i_{g+1} > h$, then by definition, u cannot have just h path sets. Hence, $i_{g+1} \le h$ and $x \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. By induction, all $x_j \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$ for all $j \in \{1,\ldots,m\}$. It follows that $x \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$ since x is adjacent to x_m and $x_m \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$ and $x_m \notin \mathcal{S}$.

2. We showed above that $\bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$ is equal to the set of all nodes $x \in \mathcal{V}$ that is either adjacent to u or has a path $x_1, \ldots, x_m$ to u where $x_i \in \mathcal{V} - \mathcal{S}$, $i \in \{1,\ldots,m\}$. Hence, we just have to show that $\mathbb{G}(u,[c])$ and $\bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u) \cup \{u\}$ contain the same nodes. Note that $u \in \mathbb{G}(u,[c])$ by definition.

   Suppose $x \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. This implies $x \in \mathcal{N}_k(u)$ for some $k \in \{1,\ldots,h\}$. Hence, x is adjacent to u or some $x_{k-1} \in \mathcal{N}_{k-1}(u) - \mathcal{S}$. That $x_{k-1} \in \mathcal{N}_{k-1}(u)$ implies x_{k-1} must be adjacent to u or to some $x_{k-2} \in \mathcal{N}_{k-2}(u) - \mathcal{S}$, and so on.

   If x is adjacent to u, then the index of x in $[c]$ must precede that of u by construction of the complete sweep. Hence $x \in \mathbb{G}(u,[c])$. If x is not adjacent to u, then there is a path $x_{k-1}, \ldots, x_1$ from x to u where $x_i \in \mathcal{N}_i(u) - \mathcal{S}$ for all $i \in \{1,\ldots,k-1\}$. By definition of the complete sweep, we know that the indices assigned to nodes in $\mathcal{N}_a(u) - \mathcal{S}$ must be greater than all the indices assigned to nodes in $\mathcal{N}_b(u) - \mathcal{S}$ if $a < b$, and the index assigned to u is greater than the index assigned to any node in $\bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. Therefore, there is a path $x_{k-1} = c_{i_{k-1}}, \ldots, x_1 = c_{i_1}$ from $x = c_{i_k}$ to $u = c_{i_0}$ where $i_k < i_{k-1} < i_{k-2} < \ldots < i_1 < i_0$. By definition, $x \in \mathbb{G}(u,[c])$.

   Now suppose $x \in \mathbb{G}(u,[c])$ and $x \neq u$. If $x \notin \mathcal{S}$, then it follows from the definition of $\mathbb{G}(u,[c])$ and part 1 above that $x \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. Suppose $x \in \mathcal{S}$. The localizability of $\mathbb{N}$ implies that each $x \in \mathcal{S}$ must have a path to u that does not include any other node in $\mathcal{S}$. Hence, it follows from part 1 that $x \in \bigcup_{i\in\{1,\ldots,h\}} \mathcal{N}_i(u)$. □

B. MOBILITY CONTROL SCHEME FOR COVERAGE

Another motivation for studying localization algorithms for sparse networks is the joint objective to improve the coverage of the network where the coverage of a network is defined to be the union of the coverage of each node in the network. A dense network may be relatively easy to localize, but its coverage is reduced because nodes are placed near to each other to guarantee the network's density while they should spread out to improve the network's coverage. To dynamically improve the network's coverage, we propose a simple but effective distributed method that guides each node's movements using only its distance measurements to its neighbors.

Consider a connected network of mobile agents each of which has a sensor with sensing radius R. We say agent j is a *neighbor* of agent i if agent j is within sensing range of agent i. Recently, in studying the coordinated dispersion of groups of mobile agents, one mobility control rule has been studied requiring that each robot moves away from its nearest neighbor. Using the notion of generalized gradient and tools from computational geometry and nonsmooth analysis, it has been proven rigorously in [12] that this rule can spread the agents out in a bounded area and each agent's location will converge exponentially fast to its equilibrium point. The efficiency, robustness and scalability of this rule has been tested using mobile robots [27]. This fully distributed rule cannot be directly used in the coverage control of mobile sensor networks due to two reasons: (i) it is assumed in [12, 27] that each node knows the exact location of its nearest neighbor all the time even when all the nodes are in constant movement; and (ii) this rule ignores consequences of breaking established links which may result in an undesirable disconnected network.

We will first modify the above rule to make it independent of the location information. Let $n_i(t)$ denote the number of neighbors of node i at time t. For any node i, when it decides to move away from its nearest neighbor after acquiring its current distance to its neighbors, it moves in a random direction with a tentative small step and measures its distances to its neighbors again. If its distance to its nearest neighbor increases, it moves in its current direction with a normal step; otherwise, it moves in the opposite direction with a normal step. Note that in this way, each node is moving in a direction of the sub-gradient of its distance to its nearest neighbor once it moves with a normal step. Note also that if node i is installed with a compass that can tell in which directions its neighbors are located, it can always know the exact gradient direction of the distance to its nearest neighbor.

Now we will consider how to preserve the network connectivity. This is achieved by two means. One is to keep each node updated about the average connectivity of the network, denoted by $\bar{n}$; and the other is to keep the nodes on the boundary fixed if its number of neighbors is below a predefined threshold. We will take the following conservative approach to make each node aware of the fact that it may be on the boundary of the network. Let $\mathrm{cov}_i(t)$ denote the *local convex hull* of agent i at time t which is the convex hull of the positions of agent i and its neighbors at time t. It can be proved [26] that if a node i is on the boundary, it must be a vertex of $\mathrm{cov}_i(t)$. Furthermore, in most of the cases, if a node i is a vertex of $\mathrm{cov}_i(t)$, it is possible for the node to find a direction in which it moves away from all its neighbors. Note again that if node i is installed with a compass, it can know it is a vertex of $\mathrm{cov}_i(t)$ if there exists a line passing through node i such that all its neighbors live in the same half-plane. The mobility control law described above is shown in Figure 18.

while $\bar{n} > 3$
  if agent i is a vertex of $\mathrm{cov}_i(t_i)$ **and** $n_i(t_i) \le 3$
    Do not move.
  else
    Move away from the nearest neighbor.

Figure 18: The mobility control rule.